Understanding the New DSP Processor Architectures

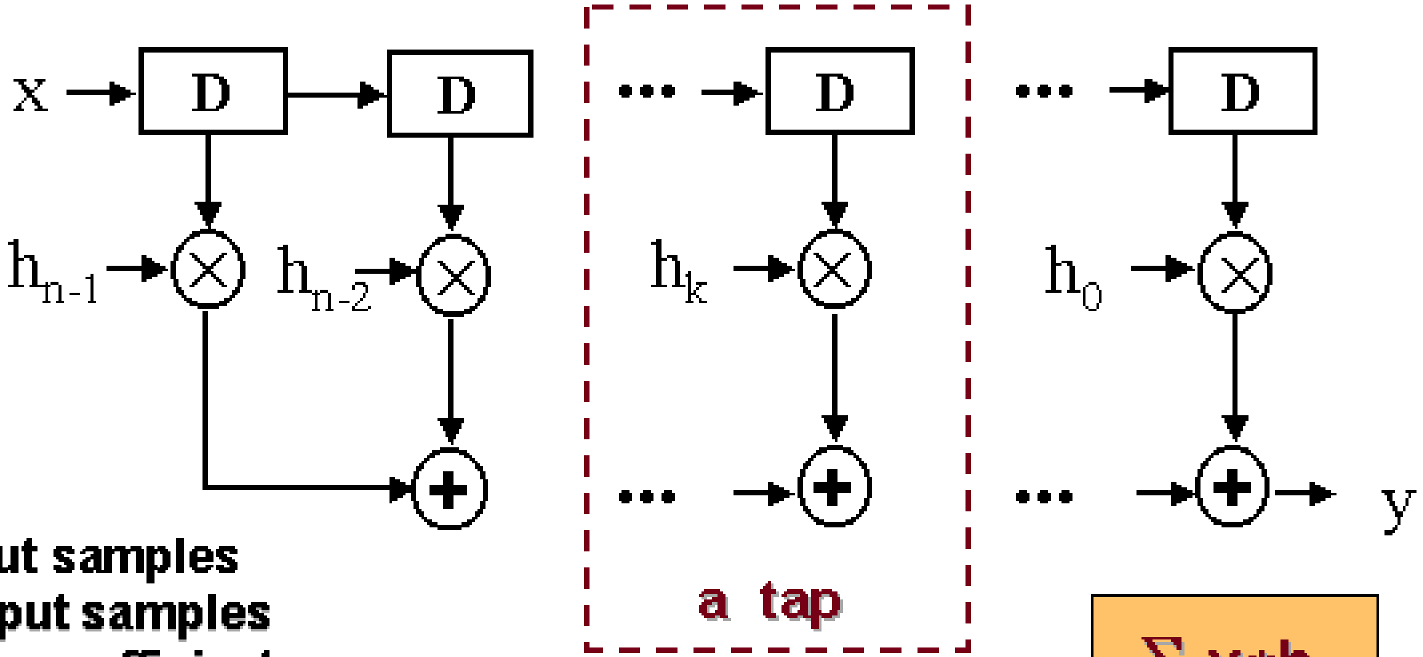
Berkeley Design Technology, Inc.
2107 Dwight Way, Second Floor
Berkeley, California U.S.A.

+1 (510) 665-1600
info@BDTI.com
http://www.BDTI.com

Outline

- DSP architectural basics
- Improved performance through increased parallelism
  - Allowing more operations per instruction
    - Enhanced conventional DSPs
    - Single-instruction, multiple-data (SIMD)
  - Issuing multiple instructions per instruction cycle
    - VLIW DSPs
    - Superscalar DSPs
- CPUs with SIMD extensions
- DSP/microcontroller hybrids

A Motivating Example:
FIR Filtering



x = input samples
y = output samples
h = filter coefficients
D = unit time delay

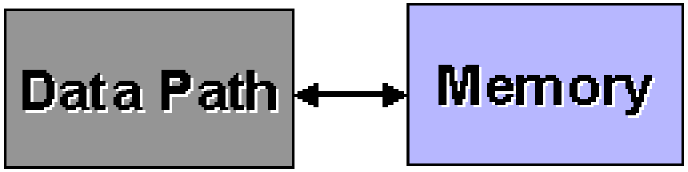
Σ x*h

Filter characteristics governed by number of taps, selection of filter coefficients.

BDTi

FIR Filter on a Typical GPP

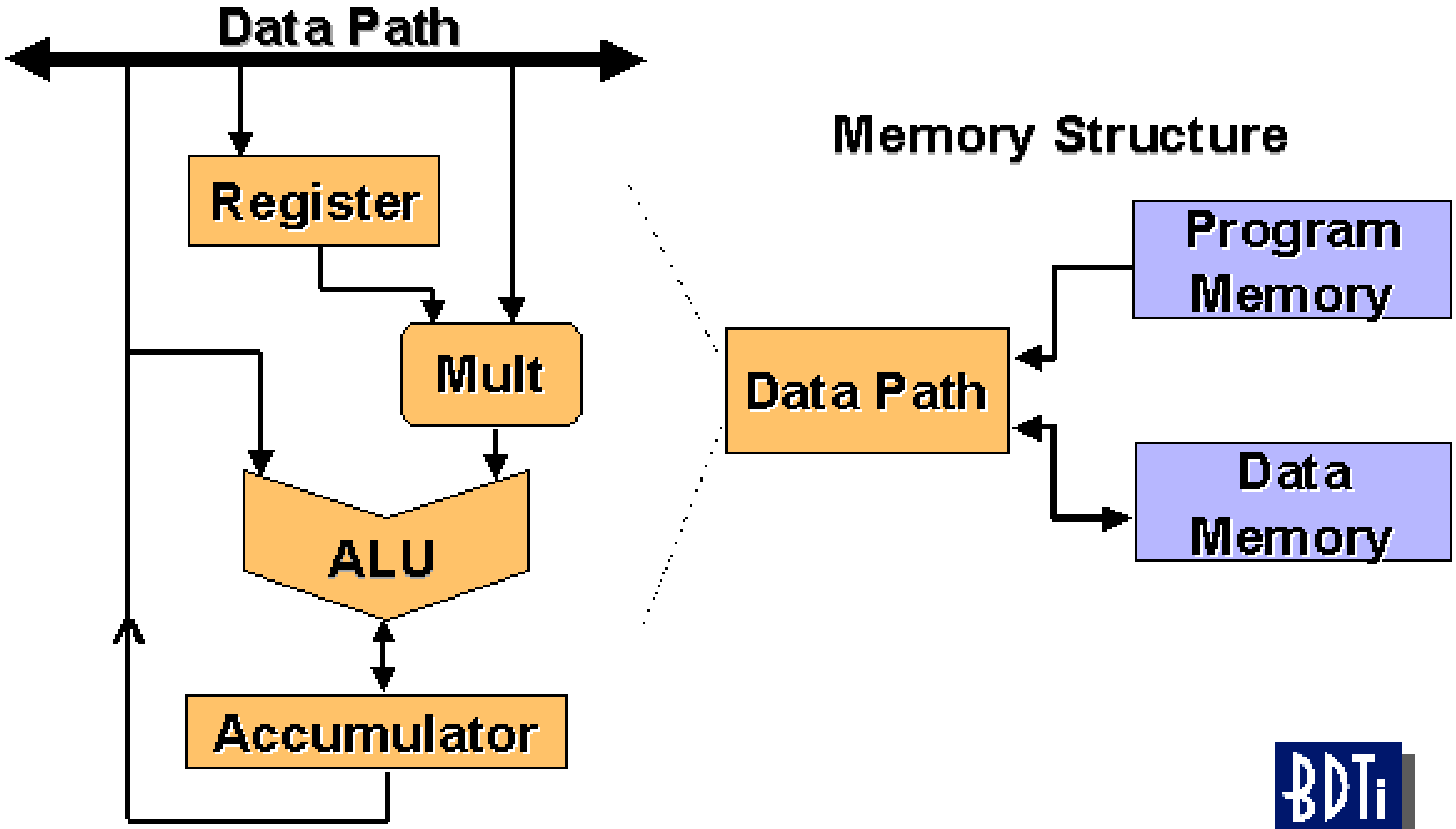
```
loop:
        mov     *r0,x0
        mov     *r1,y0
        mpy     x0,y0,a
        add     a,b
        mov     y0,*r2
        inc     r0
        inc     r1
        inc     r2
        dec     ctr
        tst     ctr
        jnz     loop
```



Problems:
- Memory bandwidth bottleneck
- Control code and addressing overhead
- Possibly slow multiply

(Computes one tap per loop iteration)

Early DSP Architecture



BDTi

FIR Filter on a Conventional DSP

```
 DO dotprod UNTIL CE;
dotprod:
  MR=MR+MX0*MY0(SS),MX0=DM(I0,M0),MY0=PM(I4,M4);
```

BDTi

Baseline: "Conventional DSPs"

- Common attributes:
  - 16- or 24-bit fixed-point (fractional), or 32-bit floating-point arithmetic
  - 16-, 24-, 32-bit instructions
  - One instruction per cycle ("single issue")
  - Complex, "compound" instructions encoding many operations
  - Highly constrained, non-orthogonal architectures

Baseline: "Conventional DSPs"

- Common attributes (cont.):
  - Dedicated addressing hardware w/ specialized addressing modes
  - Multiple-access on-chip memory architecture
  - Dedicated hardware for loops and other execution control
  - Specialized on-chip peripherals and I/O interfaces
  - Low cost, low power, low memory usage

Increasing Parallelism

- Boosting performance beyond the increases afforded by faster clock speeds requires the processor to do more work in every clock cycle. How?
- By increasing the processors' parallelism in one of the following ways:
  - Increase the number of operations that can be performed in each instruction
  - Increase the number of instructions that can be issued and executed in every cycle

More Operations Per Instruction

- How to increase the number of operations that can be performed in each instruction?
  - Add execution units (multiplier, adder, etc.)
    - Enhance the instruction set to take advantage of the additional hardware
    - Possibly, increase the instruction word width
    - Use wider buses to keep the processor fed with data
  - Add SIMD capabilities

More Instructions Per Clock Cycle

- How to increase the number of instructions that are issued and executed in every clock cycle?
  - Use VLIW techniques
  - Use superscalar techniques

- VLIW and superscalar architectures typically use simple, RISC-based instructions rather than the complex, compound instructions traditionally used in DSP processors

New Architectures for DSP

- Enhanced conventional DSPs
  - Examples: Lucent DSP16xxx, ADI ADSP-2116x
- VLIW (Very Long Instruction Word) DSPs
  - Examples: TI TMS320C6xxx, Siemens Carmel
- Superscalar DSPs
  - Example: ZSP ZSP164xx
- General-purpose processors, hybrids:
  - Examples: PowerPC with AltiVec, TriCore

BDTi

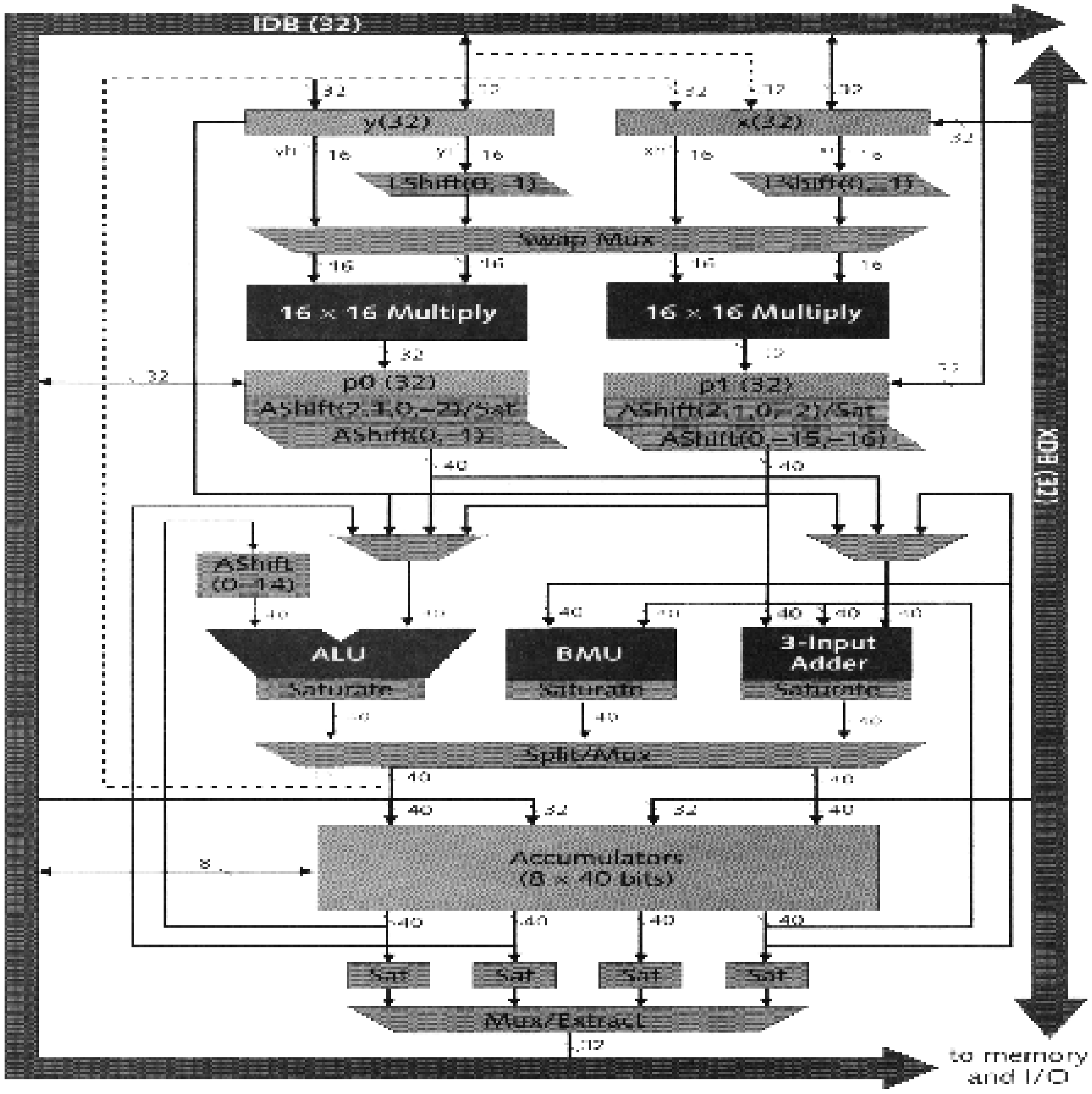
Enhanced Conventional DSPs

More parallelism via:

- Multi-operation data path
  - e.g., 2nd multiplier, adder
  - SIMD capabilities (ranging from limited to extensive)
- Highly specialized hardware in core
  - e.g., application-oriented data path operations
- Co-processors
  - Viterbi decoding, FIR filtering, etc.

Example: Lucent DSP16xxx, ADI ADSP-2116x

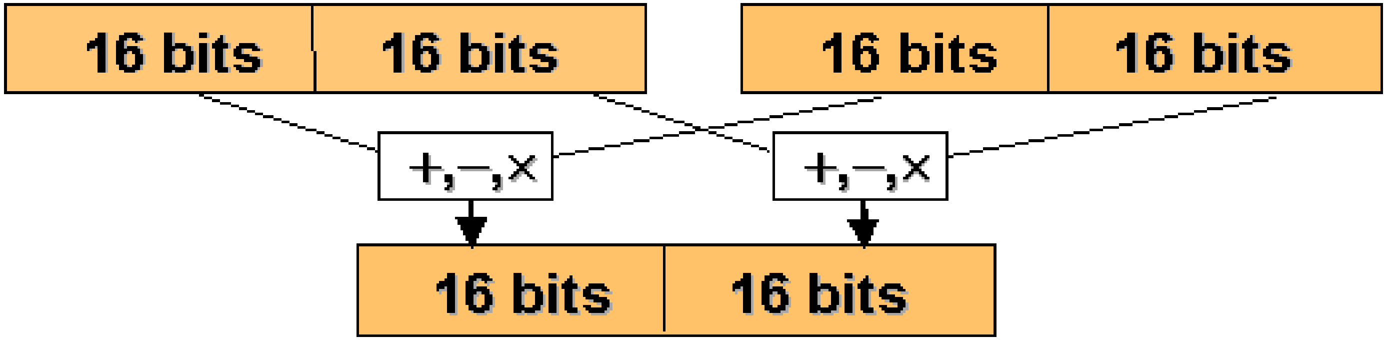
DSP16000 Data Path



FIR Filtering on the DSP16xxx

```
Do nTaps/2
 a0=a0+p0+p1 p0=xh*yh p1=xl*yl y=*r0++ x=*pt0++
```

SIMD



- Splits words into smaller chunks for parallel operations
- Some SIMD processors support multiple data widths (16-bit, 8-bit, ..)
- Examples: Lucent DSP16xxx, ADI ADSP-2116x

SIMD Extensions

- SIMD is becoming more and more common in DSP processors
  - Limited SIMD capabilities on the DSP16xxx
  - Full SIMD capabilities (enabled by dual data paths) on ADI's ADSP-2116x
- SIMD extensions for CPUs are also common. Why?
  - Make good use of existing wide resources
    - Buses, data path
  - Significantly accelerate many DSP/image/video algorithms without a radical architectural change

SIMD Challenges

- Algorithms, data organization must be amenable to data-parallel processing
  - Programmers must be creative, and sometimes pursue alternative algorithms
  - Reorganization penalties can be significant
  - Most effective on algorithms that process large blocks of data

SIMD Challenges

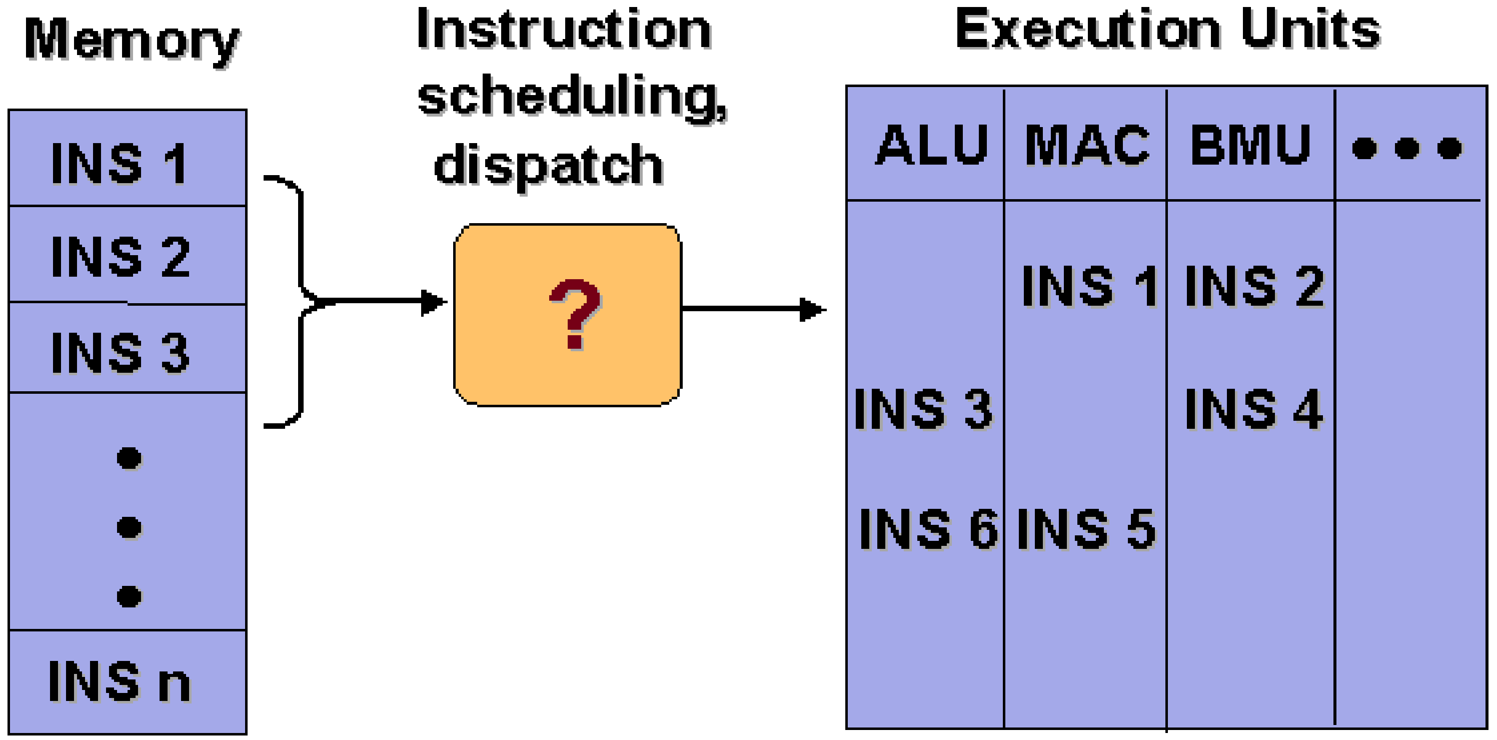
- Loss of generality
  - Each instruction processes N elements (typically $4 \leq N \leq 8$)
  - Loops often must be unrolled for speed
- High program memory usage
  - Loop unrolling
  - Re-arranging data for SIMD processing
  - Merging partial results
- Often, only fixed-point supported

Enhanced Conventional DSPs

- Advantages:
  - Allows incremental performance increases while maintaining competitive cost, power, code density
  - Compatibility is possible; similarity is likely
- Disadvantages:
  - Increasingly complex, hard-to-program architectures
  - Poor compiler targets
  - How much farther can we get with this approach?

BDTi

Superscalar vs VLIW



VLIW (Very Long Instruction Word)

Examples of current & upcoming VLIW-based architectures for DSP applications:

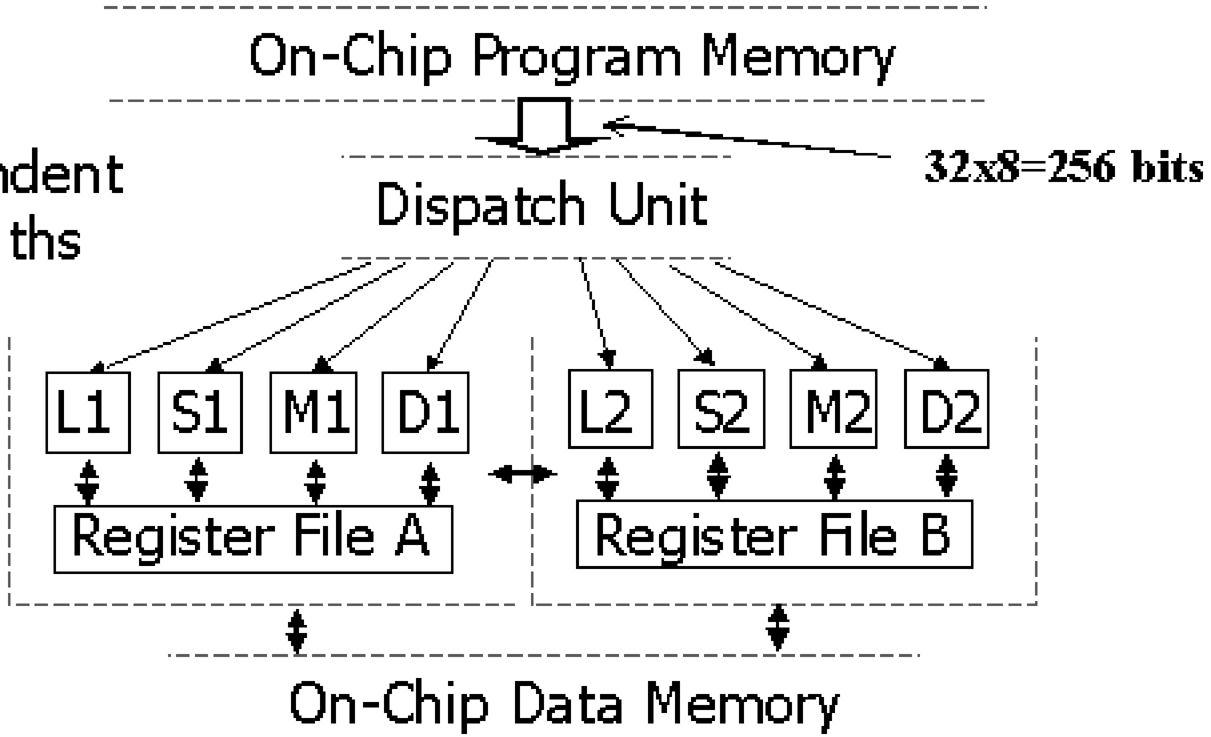
- TI TMS320C6xxx, Siemens Carmel, ADI TigerSHARC, StarCore 140

Characteristics:

- Multiple independent operations per cycle, packed into single large "instruction" or "packet"
- More regular, orthogonal, RISC-like operations
- Large, uniform register sets

Example VLIW Data Path ('C6x)

2 Independent Data Paths



FIR Filtering on the 'C6x

```
LOOP:
   ADD   .L1 A0,A3,A0
|| ADD   .L2 B1,B7,B1
|| MPYHL .M1X A2,B2,A3
|| MPYLH .M2X A2,B2,B7
|| LDW   .D2 *B4++,B2
|| LDW   .D1 *A7--,A2
|| [B0] ADD .S2 -1,B0,B0
|| [B0] B .S1 LOOP
; LOOP ends here
```

VLIW Architectures

- Advantages:
  - Increased performance
  - More regular architectures
    - Potentially easier to program, better compiler targets
  - Scalable (?)

VLIW Architectures

- Disadvantages:
  - New kinds of programmer/compiler complexity
    - Programmer (or code-generation tool) must keep track of instruction scheduling
    - Deep pipelines and long latencies can be confusing, may make peak performance elusive
  - Code size bloat
    - High program memory bandwidth requirements
  - High power consumption

Superscalar Architectures

Current superscalar architectures for DSP apps:

- ZSP ZSP164xx, Siemens TriCore (DSP/μC hybrid)

Characteristics:

- Borrow techniques from high-end CPUs
- Multiple (usually 2-4) instructions issued per instruction cycle
    - Instruction scheduling handled in hardware, not by programmer
- RISC-like instruction set
- Lots of parallelism

FIR Filtering on the ZSP164xx

```
LOOP: LDDU      R4, R14, 2
      LDDU      R8, R15, 2
      MAC2.A    R4, R8
      AGN0      LOOP
```

(All four instructions execute in a single cycle)

BDTi

Superscalar Architectures

- Advantages:
  - Large jump in performance
  - More regular architectures (potentially easier to program, better compiler targets)
  - Programmer (or code generation tool) isn't required to schedule instructions
    - But peak performance may be hard to achieve without hand-scheduling
  - Code size not increased significantly

Superscalar Architectures

- Disadvantages:
  - Energy consumption is a major challenge
  - Dynamic behavior complicates software development
    - Execution-time variability can be a hazard
    - Code optimization is challenging

Hybrid DSP/Microcontrollers

- GPPs for embedded applications are starting to address DSP needs
- Embedded GPPs typically don't have the advanced features that affect execution time predictability, so are easier to use for DSP

Hybrid DSP/Microcontrollers
Approaches

- Multiple processors on a die
  - e.g., Motorola DSP5665x
- DSP co-processor
  - e.g., ARM Piccolo
- DSP brain transplant in existing µC
  - e.g., SH-DSP
- Microcontroller tweaks to existing DSP
  - e.g., TMS320C27xx
- Totally new design
  - e.g., TriCore

Hybrid DSP/Microcontrollers
Advantages, Disadvantages

- Multiple processors on a die
  - Two entirely different instruction sets, debugging tools, etc.
  - Both cores can operate in parallel
  - No resource contention...
  - ..but probably resource duplication

Hybrid DSP/Microcontrollers
Advantages, Disadvantages

- DSP co-processor
  - May result in complicated programming model
    - Dual instruction sets
    - In ARM7/Piccolo case, possible deadlocks
  - Possible resource contention
    - e.g., Piccolo requires ARM7 to perform all data transfers
  - May allow both cores to operate in parallel

BDTi

Hybrid DSP/Microcontrollers
Advantages, Disadvantages

- DSP brain transplant in existing µC, microcontroller tweaks to existing DSP
  - Simpler programming model than dual cores
  - Constraints imposed by "legacy" architecture
- Totally new design
  - Avoids legacy constraints
  - May result in a cleaner architecture
  - Adopting a totally new architecture can be risky

Conclusions

- The variety, performance range of processors for DSP is exploding
  - Better selection, flexibility,...
  - ...but harder to choose the "best" processor
- DSPs, microcontrollers, and CPUs are swapping architectural tricks
  - CPU, μC vendors recognize the need for DSP capabilities
  - DSP, μC vendors don't want to lose sockets to each other
  - What is good in a CPU may not be good in a DSP; be careful of issues such as execution-time predictability, programmability, etc.

For More Information...

Free resources on BDTI's web site,

http://www.bdti.com

- *DSP Processors Hit the Mainstream* covers DSP architectural basics and new developments. Originally printed in IEEE Computer Magazine.
- *Evaluating DSP Processor Performance,* a white paper from BDTI.
- Numerous other BDTI article reprints, slides
- *comp.dsp* FAQ